Macintosh® Allegro Common LISP Stand Alone Application Generator

Overview

This documentation describes a function for creating stand-alone applications from Allegro CL environments. These stand-alone applications can be run on any Macintosh. They do not require the presence of a copy of Allegro CL.

Licensing Applications

An inexpensive annual license is required to ship stand-alone applications. Licenses are available from: Apple Computer Software Licensing
20525 Mariani Avenue, MS: 38I
Cupertino, CA 95014

For further information, see the shrinkwrap license included with this product

Installing the Stand-Alone Application Generator

To use the stand-alone application generator, run Allegro CL and load the file "save-application.fasl" found in the folder "Application Generator Folder". When complete, your environment includes the functionality described below.

Creating Stand-Alone Applications

The system used to create stand-alone applications is almost identical to the Allegro CL dumplisp capability. There are three differences:

- the function `save-application` is used, instead of `dumplisp`.
- the resulting image can be run without a copy of Allegro CL.
- some tools are missing from the resulting application (for example, the inspector and compiler).

Before reading further, you should be familiar with the **Dumplisp** section of the Allegro CL User's Guide.

`save-application` *pathname* `&key :compress :toplevel-function` [*Function*]

creates a stand-alone application containing the functionality of the current Lisp environment. When `save-application` is finished, Allegro CL exits to the Finder.

The syntax and semantics of `save-application` are identical to those of `dumplisp`.

pathname — a pathname specifying the directory and filename of the application to be created. If there is already a file with that name, it will be deleted before the save-application is performed.

`:compress` — if true (the default), then the heap image stored with the application is compressed as it is written. Compressed applications are smaller, but they take somewhat longer to boot.

`:toplevel-function` — the toplevel function to be set up when the application is booted. Toplevel functions are described in the **Dumplisp** section of the Allegro CL User Guide. This argument defaults to the current toplevel function. If supplied, the toplevel function must be a compiled function object; it cannot be a symbol naming a function.

The functions contained on the lists `*save-exit-functions*` and `*restore-lisp-functions*` perform the same operation as for dumped images. This process is described in the **Dumplisp** section of the Allegro CL User Guide.

If there is not enough room for the application on the selected disk, Allegro CL will error or simply quit to the Finder. Applications have a base size of about 700K.

Saving an application can take over 5 minutes (300 seconds) on 68000-based Macintoshes running from floppy disks. During this time, moving the mouse may make the cursor disappear. Don't worry, your machine hasn't crashed. Given time, Allegro CL will return to the Finder with your application created. Once an application is created, it can be run just like any other Macintosh application.

Removed Functions

The compiler and inspector are not included in stand-alone applications. Any calls to `inspect`, `compile`, or `compile-file` will simply return `nil` and have no side-effects.

Saving Space

This section provides some additional hints on saving space in stand-alone applications.

In general, files should be compiled and loaded with `*save-doc-strings*`, `*save-definitions*`, `*record-source-file*`, `*save-local-symbols*`, and `*fasl-save-local-symbols*` all set to `nil`. This prevents the saving of additional (unused) information in the application.

In general, macros (including trap macros and record definitions) do not need to be included in stand-alone applications. You can keep them out by compiling your code to fasl files (which will load your macros), exiting Lisp, restarting, and loading the fasl files (which won't need to load your macros).

The easiest way to implement efficient macro-loading behavior is with `eval-when`. If your program includes the form:

```
(eval-when (eval compile)
   (load "traps.fasl")
   (load "records.fasl")
   (load "my-macros.fasl"))
```

then, the macros will be loaded when you compile your files, but not when you load the compiled files.

You can dispose of all the predefined menus before creating your application. You can do this with the form

```
(dolist (m *default-menubar*)
   (ask m (menu-dispose)))
```

(Don't worry: the Apple menu won't be removed.)

Examples

Example stand-alone applications are found in the "Examples" folder inside the folder "Application Generator Folder".